Hindawi Publishing Corporation
Journal of Computational Medicine
Volume 2014, Article ID 526801, 11 pages
http://dx.doi.org/10.1155/2014/526801

Research Article

Genetic Algorithm Based Approach in Attribute Weighting for a Medical Data Set

Kirsi Varpa, Kati Iltanen, and Martti Juhola

Computer Science, School of Information Sciences, University of Tampere, 33014 Tampere, Finland

Correspondence should be addressed to Martti Juhola; martti.juhola@sis.uta.fi

Received 28 May 2014; Revised 30 July 2014; Accepted 6 August 2014; Published 3 September 2014

Academic Editor: Martin J. Murphy

Genetic algorithms have been utilized in many complex optimization and simulation tasks because of their powerful search method. In this research we studied whether the classification performance of the attribute weighted methods based on the nearest neighbour search can be improved when using the genetic algorithm in the evolution of attribute weighting. The attribute weights in the starting population were based on the weights set by the application area experts and machine learning methods instead of random weight setting. The genetic algorithm improved the total classification accuracy and the median true positive rate of the attribute weighted *k*-nearest neighbour method using neighbour's class-based attribute weighting. With other methods, the changes after genetic algorithm were moderate.

1. Introduction

One of the most commonly used simple classification methods is the nearest neighbour (NN) method that classifies a new case into the class of its nearest neighbour case [1]. The nearest neighbour method is an instance-based learning method that searches for the most similar case of the test case from the training data by some distance measure, usually with the Euclidean distance. A natural extension to NN is the *k*-nearest neighbour (*k*-NN) method that assigns the majority class of the *k* nearest training cases for the test case [2]. Different refinements and extensions have been proposed for *k*-NN in order to improve classification results and overcome classification problems, for example, distance-weighting of neighbours [2], extensions using properties of the data set [3], weighting of attributes [2, 4, 5], and attribute weight optimization with genetic algorithms (GA) [6–11].

Genetic algorithms [12, 13] and other evolution algorithms [14, 15] have been utilized in various complex optimization and simulation problems because of their powerful search and optimization capabilities. A search method of a genetic algorithm is a combination of directed and stochastic search and the search can be done multidirectionally because GA maintains a population of potential solutions from the search space [14]. The basics of the search method of GA underlie in natural selection and genetic inheritance [12]; individuals of the population are used in the reproduction of new solutions by means of crossover and mutation. Genetic algorithms have been used with various machine learning methods to optimize weighting properties of the method. Since our research is based on the nearest neighbour search applying machine learning methods, we concentrate on related works where GAs have been applied only with the *k*-nearest neighbour method. Kelly and Davis [6] combined the GA with a weighted *k*-nearest neighbour (*wk*-NN) method in the algorithm called GA-WKNN in order to find a single attribute weight vector that would improve the classification results of the *wk*-NN. A similar kind of approach was used in [7] where GA was combined with the *wk*-NN and a parallel processing environment in order to optimize classification of large data sets. In both studies, a set of real-valued weights for attributes to discriminate all classes of data were achieved as a result after GA runs. The study of Hussein et al. [8] showed that GA can be applied successfully in setting a real-valued weight set for 1-NN classifier but the improvement of accuracy happened at the expense of increase in processing time. Results showed that GA methods combining the *wk*-NN outperformed the basic *k*-NN [6–8]. However, a single set of weights for all classes is not always the best solution because attributes have a different effect on classes [11]. Therefore,

solutions for searching for a weight for each class and attribute have been developed. Lee et al. [9] combined the GA-based attribute weighting method with a modified k-NN, thus, forming an adaptive feature weighting method A3FW-MNN that used different sets of attribute weights for different classes. Also, Mateos-García et al. [10] assigned different weights to every attribute depending on each class in their evolutionary algorithm called Label Dependent Feature Weighting (LDFW) algorithm.

In this research we studied whether the classification performance of the attribute weighted machine learning methods based on the nearest neighbour search can be improved when using the genetic algorithm in the evolution of attribute weighting based on the experts and machine learning methods when runs were made with a medical data set. This medical data has been our test data in our previous researches [16, 17].

2. Material

In this research an otoneurological data set having 951 cases from seven different vertigo diseases (classes) (Table 1) was used. The data was collected over a decade starting from the 1990s in the Department of Otorhinolaryngology at Helsinki University Central Hospital, Finland, where experienced specialists confirmed all the diagnoses. The distribution of the disease classes is imbalanced; over one-third of the cases belong to the Menière's disease class (36.8%), whereas the smallest disease class benign recurrent vertigo has only 2.1% of the cases.

In total, the data includes 176 attributes concerning a patient's health status: occurring symptoms, medical history, and clinical findings in otoneurologic, audiologic, and imaging tests [18, 19]. Clinical testing has not been done to every patient and, therefore, there are several test results that have missing values of the attributes. Attributes with low frequencies of available values were left outside this research. After leaving out the attributes having over 35% missing values, 94 attributes remained to be used in this research: 17 quantitative (integer or real value) and 77 qualitative attributes (of which 54 were binary (yes/no), 20 were ordinal, and 3 were nominal). Genetic algorithm runs were done with the data including missing attribute values.

Table 1: The frequency distribution of vertigo disease classes.

	Disease name	Abbreviation	Frequency	%
1	Acoustic neurinoma	ANE	131	13.8
2	Benign positional vertigo	BPV	173	18.2
3	Menière's disease	MEN	350	36.8
4	Sudden deafness	SUD	47	4.9
5	Traumatic vertigo	TRA	73	7.7
6	Vestibular neuritis	VNE	157	16.5
7	Benign recurrent vertigo	BRV	20	2.1
	Total		951	100

3. Genetic Algorithm

The basic idea of the genetic algorithm is the following: in the beginning, a population of individuals is formed either randomly or with information about the application domain. Traditionally, a binary representation of the individuals has been used but in multidimensional and numerical problems real-valued representation is nowadays used [14]. In each generation, the individuals of the population are evaluated with an objective evaluation function, thus, giving the individual its fitness rate. A selection method is used to find the fittest individuals for a new population. Some individuals of the new population undergo reproduction by means of crossover and mutation. In the crossover, the information of the individuals is swapped in their corresponding elements. Mutation alters one or more elements of the individual arbitrarily. Elitism is a commonly applied survivor selection method. It keeps the current fittest individual unchanged in the population so the high-performance individuals are not lost from one generation to the next [20]. The GA can be ended after a fixed number of iterations or if no further improvement is observed after some number of generations.

We utilized the genetic algorithm in the evolution of the attribute weight values. A pseudocode of the used genetic algorithm is given in Pseudocode 1. A population contained 21 individuals that used real-valued representation instead of binary presentation because the attribute weight values were described with real-valued numbers, not just with 0 and 1. Each individual consisted of seven different attribute weight sets for 94 attributes. The individuals of the starting population were based on the weights set by the experts and machine learning methods. The starting population is defined more accurately in Section 3.1. The genetic algorithm used a roulette-wheel selection in parent selection and a uniform crossover with discrete recombination in offspring creation. The crossover was done in 80.0% probability ($p_c = 0.8$) and the crossover points were selected randomly and independently for each gene (a field on an individual). Mutation was done in 1.0% probability ($p_m = 0.01$) for the gene and it was done also in a uniform manner: a random value was drawn from the range [0, 1] which was set as a new value in the current position. In addition, elitism was used in order to keep the best individual within the population during runs. We did not want to lose the best performing weight set during the evolution. If the number of the individuals was higher than 21 in the end of the generation, a survivor selection was used. The individuals were ordered by their classification performance and the individuals with the lowest accuracy were discarded from the population. The genetic algorithm ended after 20 generations or if the best classification accuracy maintained the same during 10 successive generations. Furthermore, if all the individuals were the same in the population, the evaluation ended. The parameters used in the GA runs are described in Table 2.

The genetic algorithm runs were done separately with three different machine learning methods used in the population evaluation: with the nearest pattern method of the otoneurological expert system (ONE), with the

$$\text{data } D = \begin{bmatrix} Case_1 \\ \vdots \\ Case_{951} \end{bmatrix} = \begin{bmatrix} [c_{1,1}, \ldots, c_{1,94}] \\ \vdots \\ [c_{951,1}, \ldots, c_{951,94}] \end{bmatrix}$$

$$\text{population } Weights = \begin{bmatrix} Weight_1 \\ \vdots \\ Weight_{21} \end{bmatrix} = \begin{bmatrix} [w_{1,1,1}, \ldots, w_{1,1,94}; \ldots; w_{1,7,1}, \ldots, w_{1,7,94}] \\ \vdots \\ [w_{21,1,1}, \ldots, w_{21,1,94}; \ldots; w_{21,7,1}, \ldots, w_{21,7,94}] \end{bmatrix}$$

$population\_size = 21$
$p_c = 0.8$ //Crossover rate
$p_m = 0.01$ //Mutation rate
divide data D into 10 equally-sized subsets
for $cv\_round = 1$ to 10 **do**
    divide *training* data $D\text{-}d_{cv\_round}$ into *train* (6 subsets) and *test* (3 subsets) data
    initialize methods with *train* data:
        cwk-NN and *wk*-NN OVA: HVDM initialization
        ONE: fitness value calculation for values of attributes
    evaluate starting population *Weights* with *test* data and ONE/*cwk*-NN/*wk*-NN OVA
    while ending terms of GA are not fulfilled **do**
        //Survivor selection: Elitism
        search for the individual with the highest fitness rate from the population
        //Parent selection: Roulette-wheel selection with fitness-proportionate selection
        for each individual in the population **do**
            calculate individual's fitness proportionate rate = individual's fitness rate/sum of individuals' fitness rates
            calculate individual's cumulative fitness proportionate rate
        end for
        while nr of individuals in the mating pool is smaller than *population_size* **do**
            generate a random number r from [0, 1]
            search for the jth individual that has smaller cumulative fitness proportionate rate than r
            add the jth individual in the mating pool
        end while
        //Crossover: Uniform crossover with discrete recombination
        for each individual in the mating pool **do**
            generate a random number s from [0, 1]
            if s is smaller than p_c **then**
                add the individual in the parent pool
            else
                add the individual in the new population (offspring is a direct copy of its parent)
            end if
        end for
        while two individuals can be taken from the parent pool **do**
            if two individuals are exactly the same **then**
                add the first individual into the new population
                take new individual from the parent pool to use in the crossover
            end if
            for each disease class weight set **do**
                select the crossover points randomly
                swap information of two individuals in the corresponding crossover points (create children)
            end for
            add children in the new population
        end while
        //Mutation: Uniform mutation
        for each individual in the new population **do**
            for each gene of individual **do**
                generate a random number t from [0, 1]
                if t is smaller than p_m **then**
                    select a random value v from the range [0, 1]
                    set the value v as a new value of the gene
                end if
            end for
        end for

Pseudocode 1: Continued.

```
            evaluate children and mutated individuals in the new population with test data and
            ONE/cwk-NN/wk-NN OVA
            add the elite individual without changes into the new population
            //Survivor Selection
            if nr of individuals in the new population is larger than population_size then
                sort cases descending by their fitness rate
                discard the last cases in order to have correct nr of individuals in the population
            else if nr of individuals in the new population is smaller than population_size then
                select randomly missing cases from the old population
            end if
        end while
        initialize methods with training data D-d_cv_round:
            cwk-NN and wk-NN OVA: HVDM initialization
            ONE: fitness value calculation for values of attributes
        evaluate the individual with the highest fitness rate after GA with testing data d_cv_round and
        ONE/cwk-NN/wk-NN OVA
    end for
```

Pseudocode 1: Pseudocode of the genetic algorithm used in the evolution of the attribute weight values with 10-fold cross-validation.

Table 2: Parameters used with the genetic algorithm.

Genetic algorithm parameters	
Crossover rate	0.8
Mutation rate	0.01
Population size	21
Generation	20 (and 100 for ONE)
Elitism	Yes (1 individual)

attribute weighted k-nearest neighbour method using neighbour's class based attribute weighting (*cwk*-NN), and with the attribute weighted k-nearest neighbour method using one-versus-all the other (OVA) classifiers (*wk*-NN OVA). The evaluation methods are defined more accurately in Section 3.2. During the genetic algorithm runs, for each individual in the population its fitness rate was calculated with the method at hand; that is, the individual was evaluated against the method. Within the methods *cwk*-NN and ONE, the fitness rate for the individual was defined with a total classification accuracy (ACC) and within the *wk*-NN OVA with a true positive rate (TPR). The total classification accuracy was used with the ONE and the *cwk*-NN because all seven disease classes were classified at the same time whereas the *wk*-NN OVA concentrated on one disease class (and its weight set) at a time. During *GA wk-NN OVA* runs, it was more important to find the weight set that separated well the cases of the disease class at hand from the others than to classify the other cases also well.

The total classification accuracy showed the percentage of all correctly classified cases within the data set:

$$\mathrm{ACC} = 100\frac{t_{\mathrm{pos}}}{n_{\mathrm{cases}}}\%, \tag{1}$$

where t_{pos} was the total number of cases correctly classified within classes and n_{cases} was the total number of cases used in the classification. The true positive rate expressed the percentage of correctly inferred cases within the class as

$$\mathrm{TPR} = 100\frac{t_{\mathrm{pos}_c}}{n_{\mathrm{cases}_c}}\%, \tag{2}$$

where t_{pos_c} was the number of correctly classified cases in class c and n_{cases_c} was the number of all cases in class c. With the *cwk*-NN and *wk*-NN OVA methods, the classification performance was calculated from the seven nearest neighbour method (7-NN) results and with the ONE from the first diagnosis suggestion (ONE1). However, for disease class benign recurrent vertigo (BRV) with the *wk*-NN OVA method it was necessary to use the TPR of three nearest neighbours (3-NN) as the fitness rate because of the small size of the disease class at hand. Otherwise the TPR for classifying BRV would have always been zero. Nonetheless, if there occurred a situation where TPR of 3-NN was zero with all individuals in the starting population, a new population was created randomly and evaluated. Random new population was created at most ten times and if the TPR did not change during 10 runs, GA run was ended.

A 10-fold cross-validation (CV) [2] was used in evaluating the classification performance of the genetic algorithm. The data was randomly divided into 10 subsets of approximately equal size. The division was made in a stratified manner to ensure that the class distribution of each subset resembled the skewed class distribution of the entire data set. In the beginning, one cross-validation partition (10% of the data) was left aside to test the performance of the found best individual after genetic algorithm run. The nine cross-validation partitions (90%) were used during the training process. In order to calculate the fitness rate for each individual in the population during genetic algorithm runs, the training data was further divided into two parts: six cross-validation parts were used for training and three cross-validation parts were used for testing the current machine learning method used in the fitness rate calculation. Thus, during the genetic algorithm

run 60%–30% data division was used. After the genetic algorithm run, the individual having the highest fitness rate was declared as a result of weight combination and it was then tested with the left aside test data subset. The 10-fold cross-validation was repeated ten times. In total, there were 100 test runs per each evaluation method used in the genetic algorithm. The same cross-validation divisions were used with all the evaluation methods—that is, each method had the same training and testing sets used during the genetic algorithm runs.

3.1. Starting Population. The starting population consisted of 21 individuals. Each individual included seven different attribute weight sets (weights for 94 attributes), one set for each disease class. Instead of selecting the starting individuals at random, we decided to use good "guesses" as a starting point. Therefore, the starting individuals were based on the attribute weights defined by the domain experts (three different weight set versions) and learnt by three machine learning methods (the Scatter method [21–23] and the weighting method of the instance-based learning algorithm IB4 [24] and its variant IB1w). Based on the weight sets defined by the experts and the machine learning methods, two different modifications were created from weight sets with 50% random mutation, thus having 18 weight sets in total. In addition to these, three totally random weight sets were created into the starting population.

The weight values were computed with the machine learning methods from the imputed data set, that is, from the data set where the missing values of attributes were substituted with the class-wise modes of the qualitative and the class-wise medians of the quantitative attributes. In total, 10.1% of the values of attributes were missing in the data set. The imputation was done class-wise on the basis of the whole data prior to data division into training and testing sets. The calculation of the weights was repeated 10 times for each CV training set in the Scatter, IB4, and IB1w methods and the mean weights of the 10 repetitions were used in the classification to handle the randomness in these methods. The weights defined by the application area experts were the same for each CV training set.

The experts' weights were based on three different combinations. The first weight set included the original attribute weights defined by a group of experienced otoneurological physicians for the decision support system ONE made in the 1990s [25]. The second and the third weight sets were defined by two domain specialists during the upgrade process of the decision support system in the 2000s [16].

The Scatter method is normally used for attribute importance evaluation [21–23]. It calculates a scatter value for an attribute that expresses the attributes' power to separate classes in the data set. For attribute weighting purposes, the scatter values were calculated for each attribute in different class versus other classes' situations. In order to use the scatter values as attribute weights, it was necessary to take inverses of scatter values.

The weight calculation method of the IB4 classification method computes attribute weights independently for each class with a simple performance feedback algorithm [24]. The attribute weights of IB4 reflect the relative relevancies of the attributes in the class. The difference between IB4 and its simpler version IB1w is that IB1w saves all processed cases in its class descriptions and does not discard any cases from the class descriptions during runs. Also, the cases with poor classification records are kept in class descriptions with IB1w whereas IB4 discards these cases based on their past performance during classification.

More detailed description of the machine learning methods Scatter, IB4, and IB1w and their use in weight formation will be given in the paper [17].

In order to have different weight sets comparable to each other during the genetic algorithm runs, the attribute weights were normalized into range $[0, 1]$. The values of each weight set were divided by the highest weight value occurring in the weight calculation method at issue.

3.2. Evaluation Methods

3.2.1. Nearest Pattern Method of ONE. The first method used within the genetic algorithm to evaluate the performance of the individuals in the population was the inference mechanism of the otoneurological decision support system ONE [26]. Its inference mechanism resembles the nearest neighbour methods of pattern recognition. Instead of searching for the nearest case from the training set, it searches for the most fitting class for a new case from its knowledge base.

In the knowledge base of ONE, a pattern is given to each class that corresponds to one vertigo disease. The pattern can be considered a profile of a disease as it describes its related symptoms and signs. Each class in the knowledge base is described with a set of attributes with weight values expressing their significance for the class. In addition, a fitness value for each attribute value is given to describe how it fits the class. The fitness values for attribute values were computed on the basis of the 60% part of training data. Fitness values can have values between 0 and 100. The fitness value 0 means that the attribute value does not fit the class, whereas the fitness value 100 shows that the value fits the class perfectly. The weight values for attributes were given in the population in the GA; thus, the weight values varied from 0 to 1. The greater the weight value is, the more important the attribute is for the class.

The inference mechanism calculates scores for the classes from the weight and fitness values of the attributes. The score $S(c)$ for a class c is calculated in the following way:

$$S(c) = \frac{\sum_{a=1}^{A(c)} x(a)\, w(c,a)\, f(c,a,j)}{\sum_{a=1}^{A(c)} x(a)\, w(c,a)}, \tag{3}$$

where $A(c)$ is the number of the attributes associated with class c, $x(a)$ is 1 if the value of attribute a is known and otherwise 0, $w(c,a)$ is the weight of the attribute a for class c, and $f(c,a,j)$ is the fitness value for the value j of the attribute a for class c [26]. In the case of quantitative attributes, the fitness values are interpolated by using the attribute values in the knowledge base as interpolation points. The fitness values are altered to the range of 0 to 1 during the inference process.

In addition to the score, the minimum and maximum scores are calculated for the classes using the lowest and the highest fitness values for the attributes having missing values.

The classes are ordered primarily by the score and secondarily by the difference of the minimum and maximum score. If the classes have the same score but one class has a smaller difference between the minimum and maximum scores than the others, the class having the smallest difference is placed higher in order. If the classes have the same score and the minimum and maximum score difference, their order is selected randomly. The class having the highest score is referred to as the best diagnosis suggestion.

Some vertigo diseases resemble each other by having a similar kind of symptoms with other diseases during some phase of the disease and, in addition, some patients can actually have two (or more) vertigo diseases present concurrently [27]. Therefore, it is good to check the classification results of ONE with more than one disease suggestion. In the end, the final diagnostic choice must be made by the physician based on the information given on all alternative diseases [27].

3.2.2. Attribute Weighted k-Nearest Neighbour Method Using Neighbour's Class-Based Attribute Weighting. The other method used in the population evaluation was the attribute weighted k-nearest neighbour method using neighbour's class-based attribute weighting (*cwk*-NN). The distance measure of the basic k-nearest neighbour method [1] was expanded to take the attribute weighting into account [6]. Lee et al. [9] used a similar class-dependent attribute weighting with their modified k-nearest neighbour method where different attribute weight sets for different classes were determined with the adaptive-3FW feature weighting method. With our *cwk*-NN the attribute weighting depends on the disease class of the neighbour case. Thus, there ought to be as many attribute weights sets available as there are classes.

The distance measure used with the *cwk*-NN was the Heterogeneous Value Difference Metric (HVDM) [28] expanded with the attribute weighting. HVDM was used because it can handle both qualitative and quantitative attributes in the data set. The attribute weighted HVDM is defined as

$$\text{weighted\_HVDM}\ (x, y) = \sqrt{\sum_{a=1}^{m} w_{ca} d_a(x_a, y_a)^2}, \quad (4)$$

where m is the number of attributes, c is the disease class of the case y, w_{ca} is the weight of the attribute a in class c, and $d_a(x_a, y_a)$ is the distance between the values x_a and y_a for attribute a. The distance function $d_a(x_a, y_a)$ is defined as

$$d_a(x_a, y_a) = \begin{cases} 1, & \text{if } x \text{ or } y \text{ is unknown} \\ \text{normalized\_vdm}_a(x_a, y_a), & \text{if } a \text{ is qualitative} \\ \text{normalized\_diff}_a(x_a, y_a), & \text{otherwise.} \end{cases} \quad (5)$$

Because HVDM computes distances to qualitative and other attributes with different measurement ranges, it is necessary to scale their results into approximately the same range in order to give each attribute a similar influence on the overall distance [28]. The normalized distance to a quantitative attribute is calculated with (6):

$$\text{normalized\_diff}_a(x_a, y_a) = \frac{|x_a - y_a|}{4\sigma_a}, \quad (6)$$

where σ_a is the standard deviation of the numeric values of attribute a in the training set of the current classifier, and to a nominal attribute with (7):

$$\text{normalized\_vdm}_a(x_a, y_a) = \sqrt{\sum_{c=1}^{C} \left| \frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right|^2}, \quad (7)$$

where C is the number of output classes in the problem domain (in this case $C = 7$), $N_{a,x(y),c}$ is the number of cases in T that have a value x (or a value y) for attribute a and the output class c, and $N_{a,x(y)}$ is the number of cases in T that have a value x (or a value y) for attribute a [28]. In other words, we are calculating the conditional probabilities to have the output class c when having attribute a with the value x (or the value y).

This approach allowed modifications of all the weights at the same time.

3.2.3. Attribute Weighted k-Nearest Neighbour Method Using One-versus-All Classifiers. In addition to the neighbour's class-based attribute weighting the attribute weighted k-nearest neighbour method was tested with one-versus-all classifiers (*wk*-NN OVA). Within this method, the multiclass classification problem was converted into multiple binary classifiers—that is, the m class problem was divided into m binary problems [29]. Each binary OVA classifier was trained to separate a class from all the other classes by marking the cases of this one class as member cases and the cases of the other classes as nonmember cases in the training set.

The attribute weighted k-NN OVA is an instance-based learning method that searches for the k most similar cases (neighbours) of a new case from each classifier separately. There is one classifier per each class and each classifier gives a vote for the case being a member or nonmember of the class based on the majority class of the k neighbours. The final class of the new case is assigned from a classifier suggesting the case being a member of a class. There can occur a situation in which the new case gets more than one member of a class vote (a tie situation) or all of the classifiers vote for the other class (the case to be a nonmember of all the classes). In a tie situation the class of the new case is determined by searching for the most similar member case from the member voting classifiers. The case gets the class of the member case with the shortest distance to it. When all the classifiers vote for the case to be a nonmember, the basic 1-nearest neighbour classifier using the whole training data containing the original disease classes is employed to find the most similar case (and its class) for the new case.

The distance measure used in the *wk*-NN OVA was also the HVDM measure. The difference in the HVDM description in (4) is that the c is the class of the classifier at issue, not

Table 3: Example evaluation computation time of one population (21 individuals, one generation) in GA runs with different computers.

	Example one population evaluation time			
Computer	*GA ONE*	*GA cwk-NN*	*GA wk-NN OVA*	Specifications
C1	3 min 25 s	48 min 54 s	4 h 57 min 8 s	W7 Intel Core i7-3540M 3.00 GHz, 16 GB RAM
C2	—	49 min 53 s	6 h 59 min 16 s	I3-530 2.93 GHz, 12 GB RAM
C3	—	3 h 47 min 41 s	9 h 41 min 9 s	Q6600 2.4 GHz, 8 GB RAM
C4	—	3 h 12 min 41 s	21 h 14 min 0 s	HP ProLiant DL580 G7 server: 4* Intel Xeon X7560 2.26 GHz, 1 TB RAM
C5	—	3 h 4 min 58 s	7 h 22 min 52 s	DL785 G5 server: 8* AMD Opteron 8360 SE 2.5 GHz, 512 GB RAM
C6	—	—	10 h 34 min 55 s	Intel Core2 Duo E6750 2.66 GHz, 2 GB RAM

Table 4: The ending time of the genetic algorithm runs within different evaluation methods.

Genetic algorithm	*GA ONE*	*GA cwk-NN*	*GA wk-NN OVA*	*GA ONE100*
Ended before 20th generation [%]	75.0	18.0	82.9	39.0*
Ended on 10th generation [%]	48.0	6.0	54.9	12.0*
Ended on 20th generation [%]	25.0	82.0	17.1	61.0*

*The ending generations of the *GA ONE100* runs was examined before 100th generation, on 50th generation and on 100th generation.

the class of the case y. In addition, in (7) *wk*-NN OVA has two output classes ($C = 2$). The data in the learning set T of the classifier is divided into the member and nonmember classes.

4. Results

The results of the GA runs with ONE and *cwk*-NN as an evaluation method were the averages of the 10 times repeated 10-fold cross-validation whereas the results with the *wk*-NN OVA were the averages of the 5 times repeated 10-fold cross-validation. The 10-fold cross-validation was repeated only five times with the *GA wk-NN OVA* due to its huge computation time. For example, the evaluation of a population (21 individuals in one generation in a GA run) in one cross-validation set with the *GA ONE* lasted 3 minutes and 25 seconds, with the *GA cwk-NN* 48 minutes and 54 seconds, and with the *GA wk-NN OVA* 4 hours, 57 minutes, and 8 seconds when running the GA with the computer C1 (Table 3). With the other computers, the computation was even slower. Thus, at worst, the computation time of one cross-validation set lasting 20 generations with the computer C1 and *GA wk-NN OVA* was over four days (over 12 days with C4) assuming that within each generation all individuals were evaluated. In practice, the number of evaluated individuals varied within generations due to the crossover and the mutation. Notice that computers C4 and C5 were servers having several other users simultaneously and, thus, we had only minor part of their CPU in use. During *GA cwk-NN* and *GA wk-NN OVA* runs, the GA was run parallel in five computers, thus, having at best 11 parallel GA runs in process. *GA ONE* was run only with the computer C1.

The number of generations in the GA runs with all used evaluation methods varied from 10 to 20. In total, 75.0%, 18.0%, and 82.9% of GA runs ended before the 20th generation due to having the same best accuracy (*GA ONE* and *GA cwk-NN*) or TPR (*GA wk-NN OVA*) in 10 consecutive GA runs with ONE method, *cwk*-NN, and *wk*-NN OVA, respectively (Table 4). With the *GA wk-NN OVA*, all the GA runs with the disease classes sudden deafness, traumatic vertigo, and benign recurrent vertigo ended before the 20th generation and with the other classes from 58.0% to 88.0% of the runs. If the number of ending generation was 10, this meant that the best ACC or TPR in the population did not change at all during the GA run and, therefore, the run was ended. *GA cwk-NN* ended after 10 generations only in 6.0% of the GA runs whereas *GA ONE* and *GA wk-NN OVA* ended during the GA runs around half of runs (in 48.0% and 54.9% of runs, resp.). In the *GA wk-NN OVA* runs, this happened especially with disease class traumatic vertigo where all CV runs ended after 10 generations and with sudden deafness (96.0%) and benign recurrent vertigo (94.0%). The other disease classes ended during the *GA wk-NN OVA* runs after 10 generations from 12.0% (acoustic neurinoma) to 34.0% (vestibular neuritis) of the runs. Most of the *GA cwk-NN* runs lasted 20 generations (82.0%) whereas only a fourth of the *GA ONE* runs and 17.1% of the *GA wk-NN OVA* runs went through 20 generations.

Within the *GA wk-NN OVA* runs of the disease class benign recurrent vertigo occurred situations where the TPRs in the starting population were zero regardless of using the TPR of 3-NN instead in population evaluation. The TPR of 3-NN was used with BRV instead of 7-NN because of the small size of the disease class. The TPRs of starting individuals were zero in 30 out of 50 cross-validation sets within the *GA wk-NN OVA* run concentrating on the BRV. In this case, new starting individuals were created randomly. Random individual creation was repeated in different cross-validation sets from one to five and nine times. The *GA wk-NN OVA* run ended if the TPR of starting population stayed zero ten times. This happened in 14 (28.0%) cross-validation sets only with the disease class benign recurrent vertigo.

In order to see the effect of genetic algorithm on the population, we examined the worst and the best total classification accuracies of individuals (the attribute weight vectors) in the beginning and in the end of the genetic algorithm run. The mean worst and the mean best total accuracies and their standard deviations with GA runs using ONE and *cwk*-NN as

Table 5: The mean and its standard deviation of the best and worst total classification accuracies of individuals in the starting and ending populations occurring during different GA runs within 10 times (in *GA wk-NN OVA* 5 times) repeated 10-fold cross-validation.

Method	Population	Best accuracy [%]		Worst accuracy [%]	
		Mean	Std dev.	Mean	Std dev.
GA ONE (ONE1)	start	74.0	0.8	49.8	1.6
	end	73.8	0.7	61.4	2.8
	end 100	73.9	0.9	66.5	2.0
GA cwk-NN (7-NN)	start	63.6	1.6	27.9	2.2
	end	68.3	1.9	56.2	2.2
GA wk-NN OVA (7-NN)	start	79.2	0.5	75.3	0.5
	end	78.6	0.9	78.7	0.8

Table 6: The starting point of the genetic algorithm using ONE inference (*GA ONE*), the attribute weighted k-nearest neighbour method with neighbour's class-based attribute weighting (*GA cwk-NN*) and with OVA classifiers (*GA wk-NN OVA*) as evaluation method. The true positive rates (TPR) of seven disease classes and the total classification accuracies of the best individual from the starting population are given in percentages (%) from 10 times (five times with *GA wk-NN OVA*) repeated 10-fold cross-validation.

	Disease	ANE	BPV	MEN	SUD	TRA	VNE	BRV	Median TPR	Total accuracy
	Cases	131	173	350	47	73	157	20		951
GA ONE	ONE1	63.5	55.0	91.1	67.4	84.0	67.5	37.0	67.4	74.0
	ONE12	76.0	84.7	96.6	97.0	96.3	75.4	69.5	84.7	87.5
	ONE123	88.1	94.7	98.1	99.6	99.9	84.6	86.0	94.7	93.8
GA cwk-NN	1-NN	47.6	50.2	75.7	28.7	59.0	55.0	10.5	50.2	58.8
	3-NN	48.9	52.5	82.2	24.0	58.9	57.0	9.0	52.5	61.9
	5-NN	49.0	54.4	85.1	21.1	57.0	56.5	8.5	54.4	62.9
	7-NN	48.9	55.0	86.6	19.6	56.3	57.8	5.5	55.0	63.6
	9-NN	49.2	56.0	87.8	16.4	53.4	57.5	3.5	53.4	63.7
GA wk-NN OVA	1-NN	70.4	73.5	85.0	67.2	62.7	78.2	19.0	70.4	75.8
	3-NN	71.1	75.8	91.8	73.2	61.1	79.4	18.0	73.2	79.2
	5-NN	70.7	75.7	92.8	74.5	62.5	79.5	15.0	74.5	79.6
	7-NN	69.9	74.7	93.0	73.2	60.0	80.1	15.0	73.2	79.2
	9-NN	68.9	73.2	93.2	71.9	58.1	80.5	16.0	71.9	78.7

an evaluation method were calculated from 10 times repeated 10-fold cross-validation and with GA runs using *wk*-NN OVA from 5 times repeated 10-fold cross-validation (Table 5). The mean best accuracies stayed approximately the same with the *GA ONE*, whereas the mean best accuracy increased 4.7% with the *GA cwk-NN* and decreased 0.6% with the *GA wk-NN OVA*. The improvement can be seen from the mean worst classification accuracies: the worst accuracy occurring in the population increased during GA runs, especially with the *GA cwk-NN* (28.3%). With the *GA ONE*, the mean worst accuracy improved 11.6% when using at most 20 generations and 16.7% when using at most 100 generations. With the *GA wk-NN OVA*, the improvement was moderate (3.4%) but one must notice that its mean worst classification accuracy was already over 75% in the starting population, which was better than the mean best accuracies of the other methods.

The more detailed results of the *GA ONE*, the *GA cwk-NN*, and the *GA wk-NN OVA* runs in the beginning and in the end with the best individual occurring in the population are given in Tables 6 and 7. The true positive rates of the disease classes are shown with *GA ONE* for the first (ONE1), the first and second (ONE12), and the first, second, and third (ONE123) diagnosis suggestions of ONE and with *GA cwk-NN* and *GA wk-NN OVA* for one, three, five, seven, and nine nearest neighbours (1-NN–9-NN). During cross-validation runs in GA, the individuals were evaluated by the total classification accuracy of the ONE1 with the *GA ONE* and of the 7-NN with the *GA cwk-NN* and by the true positive rate of the 7-NN with the *GA wk-NN OVA* (except with disease class BRV that used the TPR of 3-NN). The true positive rate was used as a fitness rate with the *GA wk-NN OVA* instead of the total accuracy because it concentrated on classifying one disease class at a time whereas *GA ONE* and *GA cwk-NN* classified all seven disease classes at the same time.

Within 20 generations lasting GA, the best improvement between the start population and the end population was yielded with the *GA cwk-NN* that improved the total classification accuracies and the mean true positive rates when using one to nine nearest neighbours in the classification. Total classification accuracy of the *GA cwk-NN* rose at best 5.1%

Table 7: The end result of the genetic algorithm using ONE inference (*GA ONE*), the attribute weighted *k*-nearest neighbour method with neighbour's class-based attribute weighting (*GA cwk-NN*) and with OVA classifiers (*GA wk-NN OVA*) as evaluation method in population evaluation after at most 20 generations. The true positive rates (TPR) of seven disease classes and the total classification accuracies of the best individual in the end population are given in percentages (%) from 10 times (five times with *GA wk-NN OVA*) repeated 10-fold cross-validation.

	Disease	ANE	BPV	MEN	SUD	TRA	VNE	BRV	Median TPR	Total accuracy
	Cases	131	173	350	47	73	157	20		951
GA ONE	ONE1	63.5	55.4	90.8	66.2	83.0	68.0	31.5	66.2	73.8
	ONE12	77.0	82.7	96.4	93.6	96.2	76.2	62.0	82.7	87.0
	ONE123	87.6	92.8	98.0	98.5	99.5	84.4	84.5	92.8	93.2
GA cwk-NN	1-NN	70.2	50.0	68.4	30.6	70.0	60.3	15.0	60.3	61.1
	3-NN	70.8	53.9	78.1	27.7	72.5	62.9	14.5	62.9	65.9
	5-NN	70.5	56.1	81.5	23.2	71.9	63.8	12.0	63.8	67.4
	7-NN	69.5	56.6	84.7	21.1	71.0	63.9	8.5	63.9	68.3
	9-NN	69.0	57.5	86.6	18.1	69.7	64.1	6.0	64.1	68.8
GA wk-NN OVA	1-NN	71.5	74.1	84.6	67.2	67.1	77.8	18.0	71.5	76.2
	3-NN	71.6	75.3	91.7	74.9	66.8	78.7	16.0	74.9	79.5
	5-NN	70.4	73.6	92.2	77.0	63.6	79.2	14.0	73.6	79.1
	7-NN	70.4	71.8	92.6	77.0	59.5	79.6	13.0	71.8	78.6
	9-NN	70.5	72.4	92.7	74.9	59.7	79.6	13.0	72.4	78.7

(in 9-NN) and median TPR 10.7% (in 9-NN). The GA had a smaller effect on the results of the *GA ONE* and the *GA wk-NN OVA*. The results in the start population and in the end population stayed quite near each other. Small improvement in the mean total classification accuracy and the mean TPR can be seen with the *GA wk-NN OVA* using one or three nearest neighbours in the classification. Otherwise, the total classification accuracies decreased a bit when using the *GA ONE* and with the *GA wk-NN OVA* using five or seven nearest neighbours in the classification.

Changes within the true positive rates of disease classes compared to the start and end results varied between methods. The *GA cwk-NN* mainly increased the TPRs. During GA runs, it increased the most the TPR of acoustic neurinoma (22.6% in 1-NN) and traumatic vertigo (16.3% in 9-NN). Menière's disease was the only class where the TPR decreased (at worst −7.3% in 1-NN) during *GA cwk-NN* runs. With the *GA ONE*, the TPRs of classes mainly decreased. It decreased the most the TPR of benign recurrent vertigo (−7.5% in ONE12) and sudden deafness (−3.4% in ONE12). However, small increase in TPR can be seen with acoustic neurinoma (1.0% in ONE12) and with vestibular neuritis (0.8% with ONE12). With the *GA wk-NN OVA*, some TPRs increased and some decreased. The TPR increased the most with traumatic vertigo (5.8% in 3-NN) and sudden deafness (3.8% in 7-NN) and decreased the most with benign recurrent vertigo (−3.0% in 9-NN) and benign positional vertigo (−2.9% in 7-NN).

Because the computation time with the ONE method was so much faster than with the *k*-nearest neighbour methods, the evolution of the population with *GA ONE* runs was tested also with 100 generations in addition to the 20 generations. The ending condition was also changed: the GA run ended if the maximum accuracy stayed the same in 50 successive runs or 100 generations were run. In total, 39.0% of the *GA ONE100* runs ended before the 100th generation and within 12.0% of the runs there was no change in the best total classification accuracy during 50 generations (Table 4). The classification results of the *GA ONE100* runs are given in Table 8. The increase of generations from 20 to 100 did not affect much the mean total classification accuracy nor the mean median TPR. Within disease classes, benign recurrent vertigo suffered the most from the generation increase: its true positive rate decreased at worst −16.0% (ONE12) compared to the starting population and −9.5% (ONE123) compared to the 20th generation. The best TPR increase was achieved with acoustic neurinoma: 3.9% from the starting population and 3.6% from the 20th generation.

Table 8: The end result of the genetic algorithm using ONE inference in population evaluation after at most 100 generations. True positive rates and the total classification accuracies of the best individual in the end populationare given in percentages [%] from 10 times repeated 10-fold cross-validation.

Disease	Cases	*GA ONE 100*		
		ONE1	ONE12	ONE123
ANE	131	67.1	79.9	89.6
BPV	173	56.9	82.0	92.8
MEN	350	89.9	96.1	97.9
SUD	47	61.7	90.9	97.0
TRA	73	80.3	96.4	99.7
VNE	157	69.6	78.7	86.0
BRV	20	23.0	53.5	75.0
Median TPR		67.1	82.0	92.8
Total accuracy	951	73.9	87.3	93.5

5. Discussion and Conclusion

Genetic algorithm runs were done with three different population evaluation methods in order to see whether the classification performance of the attribute weighted methods based on the nearest neighbour search can be improved when using the genetic algorithm in the evolution of attribute weighting. The attribute weighting in the starting population was based on the weights described by the application area experts and machine learning methods instead of random weight setting. The genetic algorithm runs were done separately with the nearest pattern method of ONE (*GA ONE*), with the attribute weighted *k*-nearest neighbour method using neighbour's class-based attribute weighting (*GA cwk-NN*), and with the attribute weighted *k*-nearest neighbour method using one-versus-all classifiers (*GA wk-NN OVA*). The 10-fold cross-validation was repeated 10 times with *GA ONE* and *GA cwk-NN* and 5 times with *GA cwk-NN OVA* due to its huge computation time.

The GA runs lasted at maximum 20 generations, 10 generations if there were no change in the best classification accuracy. Most of the GA runs with *GA ONE* and *GA wk-NN OVA* ended before the 20th generation (75.0% and 82.9%, resp.) and around half (!) of the GA runs ended without a change in the best classification (ended after 10 generations; 48.0% and 54.9%, resp.). Only 18.0% of the *GA cwk-NN* runs ended before the 20th round and 6.0% after 10 generations.

The total classification accuracies and the mean true positive rates were improved within *GA cwk-NN* runs whereas with *GA ONE* and *GA wk-NN OVA* the results in the beginning and in the end population stayed quite near each other. One reason why the GA did not improve much the total classification accuracies with the *GA ONE* and the *GA wk-NN OVA* might be that the attribute weights used in the starting population were already optimized for separate disease classes. In addition, also the fitness values for ONE method can be said to be the best occurring fitness values because they were computed from the otoneurological data with the machine learning method.

Hussein et al. [8] noticed that in some applications a strict cost-benefit analysis may rule out the use of genetic algorithm optimization because of its increase in processing time (e.g., 100–150% increase in counting time compared to the basic classifier with 200 train and test cases and over 400% when using 3824 train cases and 1797 test cases with *k*-NN leave-one-out). Also, Kelly and Davis [6] admit that it can take a tremendous amount of time to find high-performance weight vectors for variably weighted machine learning methods. The results in [3] showed that the extensions of the *k*-NN yielded generally better results at the cost of speed since all extensions required a training phase. In this research, the *GA wk-NN OVA* was really time-consuming compared to *GA cwk-NN* and *GA ONE*. However, if the weight calculation needs to be done only once or quite seldom, the time issue is not that crucial, especially if it improves the performance of the method.

In this study the weights set by the experts and learnt by machine learning methods were used as a starting point. This helped a lot the search of appropriate weights but there might be different attribute weight combinations with as good or even better classification results. Therefore it would be good to test genetic algorithm also with totally random starting population and with several different parameters in offspring creation and mutation.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The first author acknowledges the support of Onni and Hilja Tuovinen Foundation, Oskar Öflund's Foundation, and Finnish Cultural Foundation, Päijät-Häme Regional fund who granted scholarships for her postgraduate studies. The authors are grateful to Docent E. Kentala, M.D., and Professor I. Pyykkö, M.D., for their help in collecting the otoneurological data and medical advice.

References

[1] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[2] T. Mitchell, *Machine Learning*, McGraw-Hill, New York, NY, USA, 1997.

[3] Z. Voulgaris and G. D. Magoulas, "Extensions of the k nearest neighbour methods for classification problems," in *Proceedings of the 26th IASTED International Conference on Artificial Intelligence and Applications (AIA '08)*, pp. 23–28, ACTA Press, Anaheim, Calif, USA, February 2008.

[4] J. M. Sotoca, J. S. Sánchez, and F. Pla, "Estimating feature weights for distance-based classification," in *Proceedings of the 3rd International Workshop on Pattern Recognition in Information Systems (PRIS '03)*, Angers, France, 2003.

[5] E. Marchiori, A. Ngom, E. Formenti, J.-K. Hao, X.-M. Zhao, and T. van Laarhoven, "Class dependent feature weighting and k-nearest neighbor classification," in *Proceedings of the 8th IAPR International Conference on Pattern Recognition in Bioinformatics (PRIB '13)*, vol. 7986 of *LNBI*, pp. 69–78, Springer, Berlin, Germany, 2013.

[6] J. D. Kelly and L. Davis, "A hybrid genetic algorithm for classification," in *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI '91)*, vol. 2, pp. 645–650, Morgan Kaufmann, San Franciso, Calif, USA, 1991.

[7] W. F. Punch, E. D. Goodman, M. Pei, L. Chia-Shun, P. Hovland, and R. Enbody, "Further research on feature selection and classification using genetic algorithms," in *Proceedings of the 5th International Conference on Genetic Algorithms (ICGA '93)*, pp. 557–564, University of Illinois, Champaign, Ill, USA, 1993.

[8] F. Hussein, N. Kharma, and R. Ward, "Genetic algorithms for feature selection and weighting, a review and study," in *Proceedings of the 6th International Conference on Document Analysis and Recognition (ICDAR '01)*, pp. 1240–1244, Seattle, Wash, USA, 2001.

[9] H. Lee, E. Kim, and M. Park, "A genetic feature weighting scheme for pattern recognition," *Integrated Computer-Aided Engineering*, vol. 14, no. 2, pp. 161–171, 2007.

[10] D. Mateos-García, J. García-Gutiérrez, and J. C. Riquelme-Santos, "Label dependent evolutionary feature weighting for remote sensing data," in *Proceedings of the 5th International Conference on Hybrid Artificial Intelligence Systems*, pp. 272–279, Springer, 2010.

[11] D. Mateos-García, J. García-Gutiérrez, and J. C. Riquelme-Santos, "On the evolutionary optimization of k-NN by label-dependent feature weighting," *Pattern Recognition Letters*, vol. 33, no. 16, pp. 2232–2238, 2012.

[12] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Mass, USA, 1989.

[13] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, Mass, USA, 1996.

[14] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Berlin, Germany, 1992.

[15] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Springer, Berlin, Germany, 2003.

[16] K. Varpa, K. Iltanen, M. Juhola et al., "Refinement of the otoneurological decision support system and its knowledge acquisition process," in *Proceedings of the 20th International Congress of the European Federation for Medical Informatics (MIE '06)*, pp. 197–202, Maastricht, The Netherlands, 2006.

[17] K. Varpa, K. Iltanen, M. Siermala, and M. Juhola, "Attribute weighting with scatter and instance-based learning methods evaluated with otoneurological data," *International Journal of Computational Medicine and Healthcare*, 2013.

[18] E. Kentala, I. Pyykkö, Y. Auramo, and M. Juhola, "Database for vertigo," *Otolaryngology—Head and Neck Surgery*, vol. 112, no. 3, pp. 383–390, 1995.

[19] K. Viikki, *Machine learning on otoneurological data: decision trees for vertigo diseases [Ph.D. thesis]*, Department of Computer Sciences, University of Tampere, Tampere, Finland, 2002, http://urn.fi/urn:isbn:951-44-5390-5.

[20] K. A. De Jong, *Analysis of the behaviour of a class of genetic adaptive systems [Ph.D. thesis]*, Computer and Communication Sciences Department, The University of Michigan, Ann Arbor, Mich, USA, 1975, http://hdl.handle.net/2027.42/4507.

[21] M. Siermala, M. Juhola, J. Laurikkala, K. Iltanen, E. Kentala, and I. Pyykkö, "Evaluation and classification of otoneurological data with new data analysis methods based on machine learning," *Information Sciences*, vol. 177, no. 9, pp. 1963–1976, 2007.

[22] M. Juhola and M. Siermala, "A scatter method for data and variable importance evaluation," *Integrated Computer-Aided Engineering*, vol. 19, no. 2, pp. 137–139, 2012.

[23] M. Juhola and M. Siermala, "Scatter Counter program and its instructions," 2014, http://www.uta.fi/sis/cis/research_groups/darg/publications/scatterCounter_2_7_eng.pdf.

[24] D. W. Aha, "Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms," *International Journal of Man-Machine Studies*, vol. 36, no. 2, pp. 267–287, 1992.

[25] E. Kentala, Y. Auramo, M. Juhola, and I. Pyykkö, "Comparison between diagnoses of human experts and a neurotologic expert system," *Annals of Otology, Rhinology and Laryngology*, vol. 107, no. 2, pp. 135–140, 1998.

[26] Y. Auramo and M. Juhola, "Modifying an expert system construction to pattern recognition solution," *Artificial Intelligence in Medicine*, vol. 8, no. 1, pp. 15–21, 1996.

[27] E. Kentala, Y. Auramo, I. Pyykkö, and M. Juhola, "Otoneurological expert system," *Annals of Otology, Rhinology and Laryngology*, vol. 105, no. 8, pp. 654–658, 1996.

[28] R. D. Wilson and T. R. Martinez, "Improved heterogeneous distance functions," *Journal of Artificial Intelligence Research*, vol. 6, pp. 1–34, 1997.

[29] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "An overview of ensemble methods for binary classifiers in multi-class problems: experimental study on one-vs-one and one-vs-all schemes," *Pattern Recognition*, vol. 44, no. 8, pp. 1761–1776, 2011.